デジタル社会推進実践ガイドブック DS-464-2

APIテクニカルガイドブック

2024年（令和6年）9月30日

デジタル庁

|  |
| --- |
| 〔キーワード〕  行政情報の流通・活用、APIエコノミー、データフォーマット、国際標準、デファクト標準、相互運用性  〔概要〕  データは公開もさることながら、活用してこそ価値を生みます。そのためには活用しやすい形式でデータが提供される必要があり、そのアクセスインタフェースとしてAPIが普及してきています。政府機関においてもAPIによるデータ提供事例が増えつつありますが、その設計・運用は各府省の個々の指針によって行われているのが現状です。  本ガイドブックでは、APIの開発に当たり技術的に考慮すべき事項や留意点を記しました。 |

改定履歴

|  |  |  |
| --- | --- | --- |
| 改定年月日 | 改定箇所 | 改訂内容 |
| 2024年9月30日 | 全体 | 主な更新点  1．はじめに  ・背景と目的、位置づけ、用語の追記  2．API設計時の留意事項  ・同期API、非同期APIの設計の追記  ・セキュリティ　通信の改ざんや盗聴防止の修正  ・APIゲートウェイの活用の追記  ・省庁間データ交換のためのAPI標準規格の追記  3．API運用時の留意事項  ・モニタリングとロギングの追記  ・SLAとSLOの設定の追記  4．API開発の進め方  ・API提供の告知の追記 |
| 2022年8月10日 | － | 政府相互運用性フレームワークに統合、改訂 |
| 2019年3月28日 | － | 初版決定 |
| 2017年8月1日 | － | β版公開 |

目次

[目次 3](#_Toc177487112)

[１ はじめに 4](#_Toc177487113)

[１.１ 背景と目的 4](#_Toc177487114)

[１.２ 適用対象 6](#_Toc177487115)

[１.３ 位置づけ 6](#_Toc177487116)

[１.４ 用語 6](#_Toc177487117)

[２ API設計時の留意事項 7](#_Toc177487118)

[２.１ URI設計及びリクエスト 7](#_Toc177487119)

[１） URI設計のポイント（WebAPI） 7](#_Toc177487120)

[２） リソース操作のメソッド（WebAPI） 8](#_Toc177487121)

[３） リソースコレクションとリソース単体に対する操作 8](#_Toc177487122)

[４） パラメータの設計（WebAPI） 8](#_Toc177487123)

[５） 同期APIと非同期APIの設計 9](#_Toc177487124)

[６） データ形式について（WebAPI） 10](#_Toc177487125)

[２.２ レスポンス 11](#_Toc177487126)

[１） サポートすべきHTTPステータス、レスポンスヘッダー 11](#_Toc177487127)

[２） 正常レスポンス（WebAPI） 14](#_Toc177487128)

[３） エラーレスポンス（WebAPI） 17](#_Toc177487129)

[４） 非同期APIのレスポンス 19](#_Toc177487130)

[５） ストリーミングAPIのレスポンス設計 20](#_Toc177487131)

[２.３ 個別データの各パラメータについて 22](#_Toc177487132)

[１） 項目設計における設計指針 23](#_Toc177487133)

[２.４ その他 25](#_Toc177487134)

[１） セキュリティ 25](#_Toc177487135)

[２） API呼び出し先と呼び出し元の処理における整合性担保 27](#_Toc177487136)

[３） APIゲートウェイの活用 27](#_Toc177487137)

[４） 省庁間データ交換のためのAPI標準規格 28](#_Toc177487138)

[３ API運用時の留意事項 29](#_Toc177487139)

[１） 利用者への告知 29](#_Toc177487140)

[２） モニタリングとロギング 30](#_Toc177487141)

[３） SLAとSLOの設定 31](#_Toc177487142)

[４ API開発の進め方 32](#_Toc177487143)

[４.１ API提供の告知 32](#_Toc177487144)

[１） ポータルの基本機能 32](#_Toc177487145)

[２） ユーザー管理とアクセス制御 33](#_Toc177487146)

[３） サポートと情報共有 33](#_Toc177487147)

[４） 使用状況と分析 33](#_Toc177487148)

[５） 更新管理と通知 33](#_Toc177487149)

[６） セキュリティとコンプライアンス 33](#_Toc177487150)

[７） フィードバックと改善 33](#_Toc177487151)

[４.２ オープン規格を活用したAPIの開発（WebAPI） 34](#_Toc177487152)

[４.３ ドキュメントの公開 39](#_Toc177487153)

[４.４ テストフォーム及びテスト環境 40](#_Toc177487154)

[４.５ サンプルプログラムの提供 41](#_Toc177487155)

[４.６ 開発者コミュニティ 41](#_Toc177487156)

[４.７ FAQ 42](#_Toc177487157)

[５ 参考情報 42](#_Toc177487158)

[５.１ URIの例 42](#_Toc177487159)

[５.２ レスポンスサンプルデータ（WebAPI） 43](#_Toc177487160)

[５.３ ドキュメント「API概要情報」の例 44](#_Toc177487161)

[参考文献一覧 46](#_Toc177487162)

# はじめに

## 背景と目的

政府は、「電子行政分野におけるオープンな利用環境整備に向けたアクションプラン」（平成26年4月25日各府省情報統括責任者（CIO）連絡会議決定）以来、行政の諸活動に関する透明性を高め、開かれた行政の実現を図ってきました。また、国民、企業等の社会・経済活動に有益な情報資源の充実に資するため、行政情報を有効活用し、より利便性の高い公共サービスの実現に向けて各種ガイドの整備と普及を推進してきました。

平成28年12月に官民データ活用推進基本法（平成28年法律第103号）、令和元年5月にデジタル手続法が制定されました。これらにより、オープンデータの公開、ワンスオンリーやワンストップなどの行政連携サービスの推進が進められています。また、令和6年6月にデジタル戦略である「デジタル社会の実現に向けた重点計画」が閣議決定され、データ連携による持続可能性が強化されることとなりました。また同月に、統合イノベーション戦略2024が閣議決定されAIの推進が重点項目として強力に推進されています。データの活用や流通にはデータに関する各種標準がさらに重要になることから、データに関するガイド類の整備が必要になっています。

データの提供は、公開することよりも、そのデータが活用され、効果を上げることに意義があります。活用を促進するためには、活用しやすい方式でデータを提供することが必要です。その観点から、Application Programming Interface（以下「API」という。）を利用したデータ提供が増えてきています。APIは、アプリケーション間のデータ共有を可能にする重要な技術です。これにより、異なるシステムやアプリケーションが効率的にデータを交換し、機能を共有できるようになります。グローバルにも広く利用され、技術や開発方法の革新が進むとともにセキュリティ対策の検討も行われています、我が国の政府機関においても、APIを取り入れてシステム構築の効率化を図るとともにイノベーションの創出を図っていく必要があります。

API導入実践ガイドブック及び本ガイドブックは、API利用者の利便性を向上し、利用を促進するため、各府省がWebサイトでAPIを提供するに当たり、共通化すべき事項や留意点などについて説明します。これらのガイドラインは、APIを通じたシステム間の相互運用性を向上させ、異なるプラットフォームや言語で開発されたアプリケーション間の連携を容易にすることを目指しています。

API導入実践ガイドブックは、主に各府省担当者向けにAPIの基本的内容（概要や意義など）を示し、本ガイドブックは、主にAPI構築担当者向けにAPIを設計・運用するに当たって考慮すべき、具体的な取組事項を示します。各府省担当者も参考として、本ガイドブックに目を通し、API構築担当者の作業の概要を把握することで、作業管理や担当者とのコミュニケーションに役立つことが期待されます。API導入実践ガイドブック、本ガイドブックの順に参照することを推奨します。

本ガイドブック及びAPI導入実践ガイドブックは、インターネットを通じて提供するWeb APIが主な対象となりますが、センサー機器、IoT等、データを提供するその他全てのAPIにおいても参考となる内容です。これらのガイドラインに従うことで、開発者は既存の機能を再利用し、新しいアプリケーションをより迅速に構築できるようになります。

なお、本ガイドブックは、遵守を求めるものではなく、データを効果的に活用するために参照し、効率的に作業することを目的としたものとして整理しました。

また、公開後も利用者のご意見を頂きつつ、継続的な改定を行ってまいります。

また、本ガイドブックの理解を助けるために、APIに係る関係者の定義を下表に示します。

表 １－１　本ガイドブックにおける、APIに係る関係者の定義

|  |  |
| --- | --- |
| APIに係る関係者 | 定義 |
| 各府省担当者 | 各府省でAPIの開発・運用担当となっている行政職員 |
| API構築担当者 | 各府省担当者から受託して、APIの開発・運用を行っている民間事業者等 |
| API利用者 | APIを用いて各種サービス等の開発・運用を行っている者 |
| サービス利用者 | APIを用いた各種サービス等を利用している者 |

## 適用対象

APIを提供する政府情報システムを対象とします。地方公共団体や民間の情報システムにおいては、政府情報システムとAPI連携する際の参考としてください。

## 位置づけ

本ガイドブックは、デジタル社会推進標準ガイドライン　DS-400 政府相互運用性フレームワーク（GIF）の一つとして位置づけられます。

## 用語

本ガイドブックにおいて使用する用語は、本ガイドブックに別段の定めがある場合を除くほか、DS-130　標準ガイドライン群用語集の例によります。

# API設計時の留意事項

WebAPIを設計する際は、RESTに基づいて設計することを推奨します。本ガイドブックにおいてRESTは、HTTPプロトコルに基づいて、URIを指定しリソースを取得する設計様式を指します。ただし、厳密な適用を必須とするものではありません。

## URI設計及びリクエスト

### URI設計のポイント（WebAPI）

以下のポイントを考慮して設計することを推奨します。

なお、以降で示すURIについては、「５　参考情報」に参考例を示します。

　・動詞や動作を表す言葉ではなく名詞を用い、複数形とする。

・抽象的な名詞は避け、API利用者がリソースの内容を把握できる単語を用いる。

・キャメルケース（区切りの文字（単語の頭文字）だけを大文字で示す表記法）ではなく、スネイクケース（単語間をアンダースコアで繋げる表記法）を推奨する。

|  |
| --- |
| 例：<https://api.example.go.jp/v1/city_libraries> |

・リソースの階層関係を明確に表現するため、必要に応じてネストされたリソースパスを使用する。

|  |
| --- |
| 例：<https://api.example.go.jp/v1/city_libraries>/{user\_id}/book/{book\_id} |

・APIのメジャーバージョンをURIに含める。バージョン番号は「v」+「整数」(v1、v2等)で表記し、バージョン番号を小数にしないこととする。APIのバージョニングにおいては、セマンティックバージョニング（Semantic Versioning）の原則を採用することを推奨する。

・ベースURIにAPI提供を表す「api」という単語を含める。apiサブドメインを推奨とする。

・リソースの操作はURIでなくHTTPメソッド（GET（データ取得）、POST （新規登録）、PUT/PATCH（更新）、DELETE（削除））で表す。

　・URIの階層が深くなる場合には、パラメータで指定する。

・URI設計は、RFC 6570「URI Template」に示されるURI Templateで指定できる形式とする。

|  |
| --- |
| 例：https://api.example.go.jp/v1/magazines/{id}.{format} |

|  |
| --- |
| 例:https://api.example.go.jp/v1/magazines.json  https://api.example.go.jp/v1/magazines.xml |

※：White House Web API Standardsを参考に作成

・URIの拡張子にて返却するデータ形式を指定することができることとする。推奨するレスポンスのデータ形式は下記「２.２　２）正常レスポンス（WebAPI）」で示す。

・運用開始後には、基本的にベースURIを変更しない。変更する場合には、前もって開発者へ通知する。詳細は「３　API運用時の留意事項」を参照されたい。

### リソース操作のメソッド（WebAPI）

URIによりリソースを指定して、HTTPメソッドを用いてデータの取得や登録など操作を行います。永続化データに求められるCRUD: Creadte（生成）、Read（読み取り）、Update（更新）、Delete（削除）操作に対応するメソッドとして一般的に普及している以下のメソッドを用いることを推奨します。

* Create: POST
* Read: GET
* Update: PUT（置換）もしくはPATCH（一部分のみ更新）
* Delete: DELETE

### リソースコレクションとリソース単体に対する操作

リソースコレクション（複数のリソースの集合）とリソース単体に対して、以下のようにHTTPメソッドをマッピングすることを推奨します。

・リソースコレクション

GET：一覧取得

POST：新規作成

・リソース単体

GET：個別取得

PUT/PATCHT：更新

DELETE：削除

### パラメータの設計（WebAPI）

以下のポイントを考慮してパラメータを設計することを推奨します。

・1つのパラメータに複数の値を指定する際には、カンマ「,」を用いる。

・ページネーションにおけるパラメータは、取得開始ページ（「offset」又は「page」）と件数（「limit」）とする。返却するデータ件数の初期値は100件以下を推奨とする。

・レスポンスデータに10件以上の返却項目を含む場合、不要なデータの返却を防ぎ、データ通信量やデータ検索負荷を削減するために、「fields」パラメータを指定し要求する項目のみ返却するようにする。

表 ２－１　WebAPIパラメータの例

|  |  |
| --- | --- |
| 名称 | 概要 |
| limit | 1回のリクエストにて返却されるデータ件数を指定する。(「per\_page」とする場合もあるが、「limit」の記述を推奨) |
| offset | 返却するデータの中で、先頭から「offset」で書かれた件数のデータを返却しないことを指定する。 |
| page | 返却してほしいデータの開始位置を指定する。「limit」と合わせて用いることが多い。 |
| since | 指定された日付以降のデータを返却する。 |
| until | 指定された日付以前のデータを返却する。 |
| sort | 指定された条件を元に並び替えて結果を返却する。 |
| encode | 文字コードを指定する。 |
| fields | 指定した項目のみを返却する。 |
| type | 返却されるフォーマットを指定する。URIにて指定することを推奨する。 |

### 同期APIと非同期APIの設計

APIの設計において、処理時間と即時性の要件に基づいて、同期APIまたは非同期APIを選択します。

同期APIはクライアントがサーバにリクエストを送信し、サーバがレスポンスを返すまで待機しますが、非同期APIはクライアントがサーバにリクエストを送信した後、サーバのレスポンスを待たずに他の作業を続行できます。

#### 同期API

以下の条件を満たす場合、同期APIを使用します。

・処理時間が短い。（1秒未満が目安）

・クライアントがすぐに結果を必要とする。

・リソースの即時的な状態変更が必要。

実装にあたっては以下の規約に従って実装を行うこととする。  
・HTTPステータスコード200 OKを使用して成功を示す。

・レスポンスボディに直接結果を含める。

・タイムアウトを適切に設定し、クライアントに通知する。（例：504 Gateway Timeout）

#### 非同期API

以下の条件を満たす場合、非同期APIを使用します。

・処理時間が長い。（1秒以上）

・バックグラウンドで処理を実行できる。

・クライアントが即時の結果を必要としない。

実装にあたっては以下の規約に従って実装を行うこととする。

・リクエスト受付時

HTTPステータスコード202 Acceptedを返す。

レスポンスヘッダーにステータス確認エンドポイントのLocationヘッダーを含めることする。

例：Location: https://api.example.go.jp/v1/status/{job\_id}を含める

・ステータス確認エンドポイント

非同期処理の実行状況を取得するエンドポイントを実装する。

例：https://api.example.go.jp/v1/status/{job\_id}

処理中のHTTPステータスコードは202 Acceptedを返却する。

完了した処理のHTTPステータスコード303 See Otherと結果のURLを含む。

・非同期処理結果取得

非同期処理の結果取得は、同期APIと同様にGETリクエストにて行う。

・非同期処理完了通知

可能であれば非同期処理の処理完了を通知できるようにWebhook等のサポートを検討する。

### データ形式について（WebAPI）

以下のポイントを考慮して、データ形式を設計することを推奨します。

・リクエストは、基本的に単純なパラメータによる値、又はカンマ区切りによる配列の受け渡しによって行うことを推奨する。

・マルチバイト文字（全角文字）については『JIS X 0213』に定義されるJIS第１水準～JIS第４水準を利用すること。

・文字コードはUTF-8を用いることとする。元データがShift\_JISなど、UTF-8ではない場合、文字を変換する際の文字化け・文字欠けに留意し、必要に応じて、文字コードの変換、文字の縮退を行うこととする。詳細は参考文献に示す「JIS縮退マップについて」を参照されたい。

・APIのリクエストデータに外字は利用しない。

・日本語のように、非英字を用いる場合にはスネイクケース（単語間をアンダースコアで繋げる表記法）を使用する。

ただし、URIのドメイン名には制約があることに留意する。（例えばスネイクケースは使用できないなど）

・パーセントエンコーディングで表現される文字列はURIに含めない。

## レスポンス

### サポートすべきHTTPステータス、レスポンスヘッダー

APIレスポンスの設計において、適切なHTTPステータスコードとレスポンスヘッダーの使用は、クライアントとサーバ間の効果的なコミュニケーションを確保する上で極めて重要です。以下に、主要なHTTPステータスコードとレスポンスヘッダーを示します。

表 ２－２　主要なHTTPステータスコード

|  |  |  |
| --- | --- | --- |
| ステータスコード | 説明 | 使用例 |
| 200 OK | リクエストが成功し、レスポンスとともに要求された情報が返される | GETリクエストの成功、更新操作の成功 |
| 201 Created | リクエストが成功し、新しいリソースが作成された | POST/PUTリクエストによる新規リソース作成 |
| 202 Accepted | リクエストは受け付けられたが、処理はまだ完了していない | 長時間処理を要する申請の受付時（非同期APIの開始時） |
| 204 No Content | リクエストが成功したが、返すべきコンテンツがない | DELETEリクエストの成功 |
| 206 Partial Content | 部分的なコンテンツが返される | 大容量データの分割ストリーミング時（ストリーミングAPI） |
| 303 See Other | 別のURLにリダイレクトする必要がある | タスク完了後、結果の取得URLへの誘導時（非同期APIの完了時） |
| 400 Bad Request | クライアントのリクエストに誤りがある | 不正なリクエストパラメータ、無効なデータ形式 |
| 401 Unauthorized | 認証が必要、または認証に失敗した | APIキーの不在、無効なアクセストークン |
| 403 Forbidden | クライアントにリソースへのアクセス権がない | 権限のないリソースへのアクセス試行 |
| 404 Not Found | 要求されたリソースが見つからない | 存在しないエンドポイントやリソースID |
| 405 Method Not Allowed | リクエストされたHTTPメソッドが許可されていない | 許可されていないHTTPメソッドの使用 |
| 408 Request Timeout | クライアントからのデータ送信がタイムアウトした | クライアントの接続が長時間無応答の場合（ストリーミングAPI） |
| 409 Conflict | リクエストがリソースの現在の状態と競合している | 同時更新による競合 |
| 422 Unprocessable Entity | リクエストは理解できたが、処理できない | バリデーションエラー |
| 429 Too Many Requests | クライアントが短時間に多すぎるリクエストを送信した | レート制限の超過 |
| 500 Internal Server Error | サーバ側で予期しないエラーが発生した | サーバ側の実行時エラー |
| 503 Service Unavailable | サービスが一時的に利用できない | メンテナンス中、過負荷状態 |

#### 表 ２－３　主要なレスポンスヘッダー

|  |  |  |
| --- | --- | --- |
| ヘッダー名 | 説明 | 使用例 |
| Access-Control-Allow-Origin | CORSポリシーを指定 | Access-Control-Allow-Origin: \* |
| Content-Type | レスポンスボディのメディアタイプ | application/json; charset=utf-8 |
| Cache-Control | キャッシュ動作の指定 | Cache-Control: max-age=3600, public |
| ETag | リソースの特定バージョンを示す識別子 | ETag: "33a64df551425fcc55e4d42a148795d9f25f89d4" |
| Link | ページネーションのための関連リンク | Link: <https://api.example.com/users?page=2>; rel="next" |
| Location | 新しく作成されたリソースのURI | Location: /users/1234 |
| Strict-Transport-SecurityHTTPS使用の強制 | Strict-Transport-SecurityHTTPS使用の強制 | max-age=31536000; includeSubDomains |
| X-Request-ID | リクエストの一意識別子 | 550e8400-e29b-41d4-a716-446655440000 |
| X-RateLimit-Limit | 期間あたりの最大リクエスト数 | X-RateLimit-Limit: 5000 |
| X-RateLimit-Remaining | 現在の期間で残りのリクエスト可能数 | X-RateLimit-Remaining: 4999 |
| X-RateLimit-Reset | レート制限のリセット時刻（Unix時間） | X-RateLimit-Reset: 1372700873 |
| X-Response-TimeAPI | レスポンス時間 | 87ms |

これらのHTTPステータスコードとレスポンスヘッダーを適切に使用することで、APIの挙動をクライアントに明確に伝えることができ、エラーハンドリングや性能最適化が容易になります。

### 正常レスポンス（WebAPI）

レスポンスデータの設計時には、以下の点に沿うことを推奨します。

なお、「５　参考情報」にレスポンスデータ例を示します。

・レスポンスデータにはタグ情報を持たず、XMLに比べてデータ量が少なく、タグ処理にかかる負荷が少ないJSON形式を推奨する。ただし、既にXMLやCSV形式で環境が整備されている場合や、扱う情報に応じて選択すること。

・また、API利用者の利便性の観点から、zip等のバイナリ形式のデータは申請書や公文書等の場合を除き、使用しない。

表 ２－４　データ形式ごとの特徴

|  |  |  |
| --- | --- | --- |
| データ形式 | 特徴 | データ形式詳細 |
| JSON | シンプルで軽量であり、データ構造はキーと値のペア形式である。JavaScriptとの親和性が高く、スキーマ定義がない。Web APIが返却するデータ形式の中で、採用例が多い。 | RFC4627「The application/json Media Type for JavaScript Object Notation (JSON)」  https://tools.ietf.org/html/rfc4627 |
| XML | WebAPIにて、JSONが主流となる前によく利用された非常に柔軟なデータ形式 | W3CExtensible Markup Language (XML) 1.0 (Fifth Edition)  http://www.w3.org/TR/xml/ |
| CSV | EXCELで利用することができるなど、API利用者の利便性が高い。 | 行と列を示すに当たり、行は改行で分けて示し、各行では列（項目）ごとに『,』（カンマ）で区切り、値を『"』（ダブルクォーテーション）で囲む。1行目が各データの属性を示すヘッダー行、2行目以降にデータを表すことが多く、データの説明等は別ファイルに示す。  RFC4180「Common Format andMIME Type for Comma-Separated Values (CSV) Files」  https://tools.ietf.org/html/rfc4180 |

・データの種類によっては、標準化された専用のデータ形式が存在するため、必要に応じて、その形式に従う。代表的なデータ形式の例を下表に示す。

表 ２－５　標準化された専用のデータ形式例

|  |  |  |
| --- | --- | --- |
| データ | データ形式 | データ形式詳細 |
| 地理空間情報 | GeoJSON | RFC7946「The GeoJSON Format」  https://tools.ietf.org/html/rfc7946 |
| KML | Open Geospatial Consortium  http://www.opengeospatial.org/standards/kmlただし、KMLは測地系をサポートしていないため、一部の地理空間データが表現することができない点に注意が必要 |
| スケジュール（イベントなど） | iCalendar | RFC5545 「Internet Calendaring and Scheduling Core Object Specification」  https://tools.ietf.org/html/rfc5545 |
| 連絡先 | vCard | RFC6350「vCard Format Specification」  https://tools.ietf.org/html/rfc6350 |

・リクエストデータと同様に、文字コードはUTF-8を用いる。必要に応じて、文字の縮退を行うこととする。詳細は参考文献に示す「JIS縮退マップについて」を参照されたい。

・APIのレスポンスデータに外字は使用しない。

・API利用者はレスポンスデータを見て開発を行うため、返却される内容で実行結果を判断することができるようにすることが望ましい。処理ステータス、処理ログ、ドキュメント、パラメータ、データ件数、実行結果データ、画面遷移などのメタデータを項目に含める。次項にメタデータの例を示す。

図 ２－１　レスポンスデータに含めるメタデータ例

|  |
| --- |
| {  "metadata": {  "title":"書籍情報取得"  "detail":"正常実行されました。"  "type":"https:// example.go.jp/docs/"  "parameter":{  "offset": 25,  "limit": 25  },  "resultset": {  "count": 227,  "offset": 25,  "limit": 25}  },  "results": []  } |

※：White House Web API Standardsを参考に作成

表 ２－６　レスポンスデータに含める代表的なメタデータ例

|  |  |
| --- | --- |
| 名称 | 概要 |
| status | 処理に対応したHTTPステータスコードをHTTPレスポンスヘッダに格納する。 |
| type | APIに関するドキュメントのURLを表示する。 |
| title | リクエストに対応するWebAPIの機能名称を表示する。 |
| detail | statusに対応する処理結果について表示する。 |
| instance | リクエストURLを記載する。 |
| parameter | WebAPIプログラムに渡されたパラメータを列挙する。 |
| resultset | 全データ件数、返却しなかった件数、返却したデータ件数など、結果セットについてのメタ情報を表示する。 |
| result | 結果データを表示する。 |

・単純なデータ形式では表現が容易ではない画面遷移などに代えて、JSON+Hypertext Application Language(HAL)形式を用いリソースへのリンクを示すことで、リソース間の遷移ができるようにする。特に手続API（一連の処理を通じ、手続に関する機能を提供するものをいう。）では、既定のフローに従って順次APIを呼び出すことが多いため、API間の繋がりを示すAPI管理等の仕組みがあることが望ましい。

図 ２－２ JSON+Hypertext Application Language(HAL)形式例

|  |
| --- |
| {  "\_links": {  "self": { "href": "/orders" },  "next": { "href": "/orders?page=2" },  "find": { "href": "/orders{?id}", "templated": true }  },  "\_embedded": {  "orders": [{  "\_links": {  "self": { "href": "/orders/123" },  "basket": { "href": "/baskets/98712" },  "customer": { "href": "/customers/7809" }  },  "total": 30000,  "currency": "JPY",  "status": "出荷中",  },]  },  "currentlyProcessing": 14,  "shippedToday": 20  } |

※：White House Web API Standardsを参考に作成

### エラーレスポンス（WebAPI）

API利用時に正常にデータのやり取りがなされなかった場合、エラー内容を示すメッセージを返します。API利用者がメッセージを見ることで、どこにどのような問題が生じているか把握できる表現にする必要があります。

RFC 7807「Problem Details for HTTP APIs」で示されるように、メッセージは以下のデータ項目を含むことを推奨します。

表 ２－７　エラーレスポンスに含むデータ項目

|  |  |
| --- | --- |
| 名称 | 概要 |
| status | 処理に対応したHTTPステータスコードをHTTPレスポンスヘッダに格納する。 |
| type | エラーの種別を示すURIを格納する。参照先のURIでは、API利用者がエラー種別を理解できるドキュメントを用意していることが望ましい。該当するURIがない時には、”about:brank”と入力する。 |
| title | エラーの名称を表示する。 |
| detail | API利用者がどこに問題があるか理解できるよう、エラーの説明文を格納する。 |
| instance | エラーが発生した場合のURIを記載する。 |

図 ２－３　エラーレスポンス例（JSON形式）

|  |
| --- |
| HTTP/1.1 403 Forbidden  Content-Type: application/problem+json  Content-Language: ja  {  "type": "https://example.com/probs/out-of-credit",  "title": "口座残高が十分でありません。",  "detail": "口座残高は3000円です。5000円の支払いが必要です。",  "instance": "/account/12345/msgs/abc",  "balance": 3000,  "accounts": ["/account/12345",  "/account/67890"]  } |

※上記例では、推奨項目にbalance、accountsを追加している。

※：RFC 7807「Problem Details for HTTP APIs」を参考に作成

メッセージに含むコードは、既存のHTTPステータスコードだけでなく、より具体的なエラー情報を示すためにアプリケーション独自のエラーコードを設定することもあります。

ここでは、既存のHTTPステータスコードのうち、APIにおいて参照することの多いコードの例を示します。

表 ２－８　 HTTPステータスコード例

|  |  |  |
| --- | --- | --- |
| 番号 | 名称 | 概要 |
| 200 | OK | リクエスト成功 |
| 400 | Bad Request | リクエストデータに問題あり（パラメータの内容や文字コードが誤っているなど） |
| 401 | Unauthorized | 認証されていない。 |
| 403 | Forbidden | アクセスを拒否された（利用制限を越えているなど）。 |
| 404 | Not Found | URLにリソースが存在しない。 |
| 500 | Internal Server Error | サーバ側での処理中にエラーが発生 |
| 503 | Service unavailable | サービス利用不可（サーバがメンテナンス中など） |

### 非同期APIのレスポンス

#### リクエスト受付時のレスポンス

・HTTPステータスコード202（Accepted）を返す。

・レスポンスボディに、タスクの識別子と状態確認用のURIを含める。

図 ２－４　リクエスト受付時のレスポンス例

|  |
| --- |
| {  "task\_id": "123e4567-e89b-12d3-a456-426614174000",  "status": "processing",  "status\_url": "/api/v1/tasks/123e4567-e89b-12d3-a456-426614174000" } |

#### 状態確認エンドポイント

・タスクの現在の状態を返す。

・完了した場合は、結果へのリンクや直接的な結果を含める。

図 ２－５　状態確認エンドポイント例

|  |
| --- |
| {  "task\_id": "123e4567-e89b-12d3-a456-426614174000",  "status": "completed",  "result\_url": "/api/v1/results/987fgh-e89b-12d3-a456-426614174000"  } |

#### エラー処理

・処理中にエラーが発生した場合、適切なエラーコードとメッセージを返す。

図 ２－６　エラー処理例

|  |
| --- |
| {  "task\_id": "123e4567-e89b-12d3-a456-426614174000",  "status": "failed",  "error": {  "code": "PROC\_ERROR",  "message": "データ処理中にエラーが発生しました。"  }  } |

#### タイムアウト処理

・長時間実行されるタスクには適切なタイムアウト設定を行い、タイムアウト時の挙動を定義する。

#### タスク完了通知

・可能であれば、タスク完了時にクライアントにタスク完了通知するWebhook等のサポートを検討する。

### ストリーミングAPIのレスポンス設計

ストリーミングAPIは、リアルタイムでデータを送信する場合や、大量のデータを段階的に送信する場合に使用されます。以下のポイントを考慮して設計することを推奨します。

#### コネクション確立

・HTTP/1.1の場合、Transfer-Encoding: chunkedヘッダーを使用する。

・WebSocketを使用する場合は、適切なハンドシェイクプロセスを実装する。

#### データフォーマット

・各データチャンクが独立して解析可能な形式を使用する。

・JSON Lines形式（各行が有効なJSON）の使用を推奨する。

図 ２－７　データフォーマット例

|  |
| --- |
| {"id": 1, "data": "チャンク1"}  {"id": 2, "data": "チャンク2"}  {"id": 3, "data": "チャンク3"} |

#### エラー処理

・ストリーム中にエラーが発生した場合、エラーメッセージを特別なフォーマットで送信する。

図 ２－８　エラー処理例

|  |
| --- |
| {"error": true, "code": "STREAM\_ERROR", "message": "データ送信中にエラーが発生しました。"} |

#### 接続管理

・クライアントの接続が切れた場合の再接続メカニズムを提供する。

・サーバ側で適切なタイムアウト処理を実装する。

#### バックプレッシャー処理

・クライアントがデータを処理しきれない場合のバックプレッシャーメカニズムを実装する。

#### セッション終了

・ストリームの終了を明示的に示す方法を提供する（例：特別な終了メッセージの送信）。

#### サンプルレート制御

・必要に応じて、クライアントがデータの送信レートを制御できるメカニズムを提供する。

## 個別データの各パラメータについて

データの利活用を促すためには、他のデータとの統合や掛け合わせ（結合）がしやすいことが重要であり、データ項目の名称、形式、意味定義が共通で一意性が確保できていることが望ましいと言えます。

そのため、レスポンスデータ等は国際標準やデファクト標準等に従って、データ項目名及び内容を設定することを推奨します。

分野を問わず使用される基本的なデータ（時刻、住所、電話番号等）については、政府相互運用性フレームワーク（GIF）として推奨する書式等を公開しています。

下表に推奨する表記方法を示します。

表 ２－９　推奨する表記方法例

|  |  |  |
| --- | --- | --- |
| 内容 | 表記方法 | 表記例 |
| 日時 | ISO 8601に準拠。  日付のみであれば、全て半角でYYYY-MM-DD  日時（日本時間）の場合、日付と時間の間にTを挟み、全て半角でYYYY-MM-DDTHH:mm:ss+09:00と記載する。日本ではUTCから9時間の時差があるため、+0900とする。 | 日付のみであれば、『2017-02-06』、日時（日本時間）であれば、『2017-02-06T13:50:40+0900』で表す。 |
| 言語 | ISO 639-1に準拠。  日本語を必須、英語については推奨とする。多くの言語に対応する必要がある場合には、ISO639-2やISO639-3などを検討する。 | 日本語の場合『ja』、英語の場合『en』で表す。 |
| 貨幣 | ISO 4217に準拠。 | 日本円の場合『JPY』、アメリカドルの場合『USD』で表す。 |
| 性別 | ISO 5218に準拠。 | 男性の場合『1』、女性は『2』、それ以外は『9』、不明な場合には『0』で表す。 |
| 国 | ISO 3166-1 alpha-2に準拠。 | 日本の場合『JP』、米国の場合『US』で表す。 |
| 都道府県 | JIS X 0401に準拠。 | 東京都の場合は『13』で表す。 |
| 市区町村 | JIS X 0402に準拠。 | 東京都千代田区の場合は『101』で表す。 |
| 公衆向け警報において危険の深刻度を表すカラーコード | ISO 22324に準拠。 | 危険を赤色、注意を黄色、安全を緑色で表す。 |
| 地理座標 | 地理空間情報（緯度経度データを含む）はGeoJSON形式。  測地系はJGD 2011/(B,L)を基本とし、緯度経度のデータとあわせて測地系を別項目で記述。 | 東京都千代田区永田町2丁目3-1は、JGD 2011/(B,L)で  『35.676097, 139.744879』と表す。 |

ただし、標準では必須・推奨・任意などの推奨度合が示されているケースが多く、全ての項目を一律に用意する必要はありません。標準の中から作成するAPIの仕様に沿う、必要な項目だけを選択して活用することや標準に準拠しているデータ項目と、追加したデータ項目が分かるように、API仕様書などドキュメントで示すことを推奨します。

以下にデータ項目の例を示します。 表 ２－７以外の表記法に関しては、共通語彙基盤・文字情報基盤を参照願います。詳細は巻末参考文献に示す共通語彙基盤及び文字情報基盤のWebサイト（https://imi.go.jp/goi/）で公開しています。

その他、データ項目を設計する際に留意すべき点を示します（以下一覧はAPI導入実践ガイドブック「３．１ 設計の手順　４) データ項目定義」の再掲です）。

### 項目設計における設計指針

#### データ項目名

データ項目名は分かりやすい名称を使用することとし、名詞を基本として複数形で記載することを推奨します。また、設計上の支障がなければ、単語間をアンダースコアで繋げるスネイクケース（例：city\_libraries）を推奨します。適切でない英語名をデータ項目名に付与することやデータ項目名の意味定義が曖昧な場合、他のサービスとの連携時に誤解を生み、データ不正や不備が発生する可能性があります。

#### 入力規約

日時など国際標準や業界固有のデファクト標準等で、表現、記述方法が複数存在するものや法人名のヨミガナのように書き方のルールが必要なものについては、API仕様書の中で入力規約として記述します。

#### 個別記述と列挙

データ項目内に同種の内容を複数示す場合、複数の項目を分け個別の項目として記述するか、一つのデータ項目内に列挙するかを検討することになります。例えば、施設の利用可能日を記述する場合、以下のように記述する方法があります。

・曜日ごとに個別の項目で記述する方法

月：○、火：×、水：○・・・

・列挙して記述する方法

利用可能日：月,水,・・・

基本的に、一つずつの要素（上記例では曜日）の属性（上記例では利用可否○/×）を示す場合には個別の記述とし、条件に合致するものに何があるか（上記例では利用可能な曜日は何か）を示す場合には列挙する記述を使用します。その他、扱うデータ項目ごとに変更への柔軟性や項目数などを勘案して決定します。

#### コード（分類体系）

データ項目によっては、分類や基本属性を示すためにコードを活用することがあります。

コードを使う場合は各種標準コードがあるか調べ、できる限りその標準コードを利用します。独自コードを設計する場合には、別冊のコード（分類体系）設計・運用実践ガイドブックを参照してください。

#### 構造化

データを構造化し、共通的に利用していくことで、API利用者の利便性、各府省担当者やAPI構築担当者の業務の効率性を高めることができます。

データの構造化については、別冊の「マスターデータ等基本データ導入実践ガイドブック」を参照してください。

## その他

### セキュリティ

(1) APIにおけるセキュリティ対策（WebAPI）

APIを設けるということは、外部から常時アクセスできる入口を作ることとなるため、セキュリティリスクを持つことになります。そのため、セキュリティリスクへの対策が極めて重要となります。一般的なREST APIにおいて必要なセキュリティ対策例を下表に示します。

表 ２－１０　REST APIにおけるセキュリティ対策例

|  |  |  |
| --- | --- | --- |
| 観点 | 対策事項 | 内容など |
| 通信の改ざんや盗聴防止 | TLS(SSL)サポート（HTTPS通信）による暗号化 | Webサイトと同様に、通信データの暗号化により、送受データの信頼性担保、情報漏えいの防止が必須であり、必ずTLS(SSL)サポートの対応をする必要がある。最新のTLS 1.3プロトコルを採用し、すべての通信でHTTPSを強制する。HTTP Strict Transport Security (HSTS)ヘッダーを実装し、ブラウザに常にHTTPSを使用するよう指示する。証明書は信頼できる認証局から取得し、適切に管理する。 機密データはエンドツーエンドで暗号化し、転送中および保存時のデータを保護する。 |
| 情報漏えい防止 | APIキー又はOpenID ConnectによるAPI利用者認証 | API利用者認証を強く推奨する。少なくともAPIキーによる実装以上の認証レベルを担保すること。API利用者情報の登録を行い、登録を終えた者に対してAPIキーを付与する。  ただし、APIキーはコピーされアクセスされる可能性もあるため、強固な対策ではないことに留意する。そのため、例えば、パラメータにAPIキーを含めないこと、また発行後不要になったAPIキーを削除することなどの対応が必要となる。  手続APIのように、行政手続を行う場合や個人情報を扱う場合など、個人認証を正確に行う必要がある際には、OpenID Connectの認証も行うことを推奨する。（OpenID Foundationにより適切に実装されていると認められたソフトウェアは、次のWebページにリストとして示されている。http://openid.net/certification/）。更に極めて守秘性の高い情報についての認証が必要な場合には、OpenID Connectの手法のうち、Hybrid Flowを採用することを推奨する。詳しくは、巻末参考文献のOpenID Japanを参照する。 |
| サーバ負荷の管理 | 利用制限 | 使いやすいAPIは攻撃者にとってもアクセスしやすく、DoSアタック(Denial of Service attack) を仕掛けられるなど攻撃の標的にされる可能性が高くなる。  対策として、利用申請及び認証によりAPI利用者単位のAPIコール数を把握し、利用制限をかけることが挙げられる。API利用者ごとに、アクセス時間・回数又は取得データによる制限を行う。例えば、全件ダウンロードの禁止、1日のダウンロード件数上限、1日のアクセス回数上限などがある。大量取得を発見すれば、そのような行為を行ったAPI利用者へ通告することも可能である。ただし、制限を厳しくし過ぎると、利便性を損なうため、利用者の立場から、適切に利用制限を行う必要がある。  また、手続APIでは、サービス利用者個々人に関するデータを送受するため、セキュリティ対策は、より強固とする必要がある。例えば、申請処理を扱うAPIで対策を怠れば、サービス利用者個々人の申請情報が書き換えられたり、盗み見られたりすることに繋がる。 |
| キャッシュ | 同一のGETリクエストに対して、キャッシュからレスポンスを返す仕様とする。 |
| クロスドメインの通信許可 | CORS対応 | 通常Webブラウザでは、クロスサイトスクリプティングを防止すべく、クロスドメインの通信を行わない機能が組み込まれている。Webサービスの利用を通じてAPIを呼び出す場合、Webサービスと別のドメインにアクセスこととなるため、クロスドメインの通信を許可するCORS（Cross Origin Resource Sharing）対応が必要となる。詳細は、巻末参考文献に示すW3CCORS勧告を参照すること。 |

特にサーバ負荷対策では、API利用者の使い方に応じて条件を付加するなど個々の対応が必要な場合があり、利用規約にその対応事項を記載することになります。例えば、同一のGETリクエストを繰り返し行う場合にはローカルにキャッシュすること、大量データを要求するケースではAPIからでなくWebサイトからファイルとして直接ダウンロードすることなどの留意事項を記載します。

(2) その他代表的なセキュリティ対策

他に考慮すべきセキュリティリスクとして、DoSアタック、SQLインジェクション、クロスサイトリクエストフォージェリ、クロスサイトスクリプティング、クリックジャギングなどが挙げられ、必要に応じて対策を講じる必要があります。詳細は、参考文献に示すIPA「ウェブサイトにおける脆弱性解説」、OWASP「REST Security Cheat Sheet」などを参照願います。

### API呼び出し先と呼び出し元の処理における整合性担保

手続APIは、API呼び出し先の処理と呼び出し元の処理において、どちらも正常に処理が終了したかどうか整合を取る必要があります。例えば、API呼び出し先が正常に処理を終了することができたとしても、呼び出し元の処理がその後、異常終了した場合には、互いのデータの整合が取れない事態が発生する可能性があります。申請完了の通知を受けたにも関わらず実際は申請ができていないケース、申請途中でエラーになったはずなのに正常に申請ができている事態などが発生し得ます。

このような事態を避けるため、通常は一貫性の確保できるトランザクション処理を構築して対処する必要がありますが、APIを介したトランザクション処理の管理は複雑となる場合が多く、場合によってはマニュアルでの対処を許容するなど、整合性に対する割り切りを考慮した簡易な設計を目指すことを推奨します。

### APIゲートウェイの活用

APIゲートウェイは、複数のAPIを統合的に管理し、セキュリティ強化やトラフィック制御、モニタリングなどを実施する重要なコンポーネントです。政府機関のAPIにおいても、APIゲートウェイの活用を強く推奨します。APIゲートウェイを導入することで、以下のようなメリットが得られます：

・セキュリティの強化：APIゲートウェイは、認証、認可、暗号化などのセキュリティ機能を一元的に管理します。これにより、各APIの個別実装の負担が軽減され、セキュリティポリシーの統一的な適用が可能になります。

・トラフィック制御：APIゲートウェイを通じて、レート制限やスロットリングを実装することで、DoS攻撃からのAPIの保護や、リソースの公平な分配が可能になります。

・モニタリングと分析：APIの使用状況、パフォーマンス、エラー率などを集中的に監視し、分析することができます。これにより、APIの改善点の特定や、ユーザーのニーズの理解が容易になります。

・バージョン管理：APIゲートウェイを通じて、異なるバージョンのAPIを並行して運用し、クライアントの移行を円滑に行うことができます。

・プロトコル変換：レガシーシステムとの連携や、異なるプロトコルを使用するクライアントへの対応が容易になります。

ただし、APIゲートウェイには導入や追加のコストの発生、最適な設計が必要になるなど導入する上で注意点があるので、それらを考慮した上で活用することが重要です。

### 省庁間データ交換のためのAPI標準規格

政府機関間でのデータ交換を効率化し、相互運用性を高めるために、省庁間データ交換のためのAPI標準規格の採用を推進します。この標準規格は、「政府相互運用性フレームワーク（GIF）」の一部として位置づけられ、以下の要素を含みます

・共通データモデル：人物、組織、住所、日時などの基本的なデータ項目について、共通のデータモデルを定義します。これにより、異なる省庁間でのデータ解釈の一貫性が確保されます。

・標準API仕様：RESTful APIの設計原則に基づき、リソース命名規則、HTTPメソッドの使用、エラーハンドリングなどの標準的な仕様を定義します。

・セキュリティ要件：標準的な認証・認可プロトコルの使用を規定し、データ交換時のセキュリティを確保します。（例えば認証プロトコルであればOpenID Connectなど）

・メタデータ規格：APIの機能や使用方法を記述するためのメタデータ形式を標準化し、API探索とドキュメンテーションを容易にします。

・サービスレベル合意（SLA）：省庁間でのAPI利用に関する性能、可用性、サポートレベルなどの基準を定めます。

この標準規格に準拠することで、以下のような利点が得られます。

・省庁間のシステム連携が容易になり、新しいサービスの迅速な開発が可能になります。

・データの一貫性が確保され、データ品質が向上します。

・共通の規格を使用することで、開発・運用コストが削減されます。

・異なる省庁のデータを組み合わせた新しい価値創造が促進されます。

各省庁がAPIを開発・提供する際は、この標準規格に準拠することを強く推奨します。また、既存のAPIについても、段階的にこの標準規格への準拠を進めることが望ましいです。

# API運用時の留意事項

### 利用者への告知

API利用者向けの重要情報（API仕様変更など）は、APIを利用しているサービスに影響を与える恐れがあるため、確実にAPI利用者へ通知する必要があります。下表のような対応を取ることを推奨します。

表 ３－１　重要情報の通知方法

|  |  |
| --- | --- |
| 通知方法 | 備考 |
| HP上の告知 | APIを紹介するサイトや開発者専用ページなど |
| 開発者への直接メール | 利用申請を課すことでAPI利用者情報の登録を行い、連絡手段を確保することが重要となる。 |
| 開発者への説明会の実施 | 新規にAPIを公開するときや、仕様や規約の変更をしたとき等、重要な情報を伝える必要がある場合等に実施。ただしOpen API等で提供している場合、利用者が多岐に渡る可能性があるため、説明会の開催は現実的ではなく非推奨。コミュニティへの投稿や告知の徹底が推奨される。 |
| 開発者コミュニティへの投稿 | 当該APIの開発者コミュニティに投稿する。 |
| 開発者向けブログへの投稿 | RSS機能をブログに付与していることが望ましい。 |

なお、API利用者側で改修、そのエンドユーザへの通知等の対応に時間を要する場合もあるため、通知から変更実施までには十分な移行期間を考慮する必要があります。

### モニタリングとロギング

APIの安定した運用と継続的な改善のために、適切なモニタリングとロギングは不可欠です。以下の点に留意してモニタリングとロギングを実施してください。

#### パフォーマンスモニタリング

レスポンスタイム、スループット、エラー率などの主要指標を常時監視します。

ピーク時の負荷状況を把握し、必要に応じてスケーリングを行います。

#### 可用性モニタリング

APIのエンドポイントごとの稼働状況を監視します。

計画外のダウンタイムを即時に検知できる仕組みを整えます。

#### セキュリティモニタリング

異常なアクセスパターンや不正利用の兆候を監視します。

セキュリティインシデントの早期発見と対応を可能にします。

#### 利用状況分析

API利用者ごとの利用頻度、パターンを分析します。

人気のあるエンドポイントや機能を特定し、改善に活かします。

#### ログ管理

アクセスログ、エラーログ、セキュリティログを適切に保管します。

ログの保存期間は法令や組織のポリシーに従って設定します。

個人情報や機密情報の扱いに十分注意し、必要に応じて匿名化や暗号化を行います。

#### アラート設定

リクエストの処理に要する時間など重要な指標が閾値を超えた場合に、即時通知される仕組みを整えます。

アラートの優先度を適切に設定し、運用チームの負担を最適化します。

#### 可視化とレポーティング

モニタリング結果をダッシボードなどで可視化し、運用状況を一目で把握できるようにします。

定期的なレポートを作成し、APIの運用状況や改善点を関係者と共有します。

### SLAとSLOの設定

APIの信頼性と品質を確保するために、サービスレベル指標（SLI）を定めた上で、適切なサービスレベル合意（SLA）とサービスレベル目標（SLO）を設定することが重要です。

#### SLAの設定

・API利用者との間で、サービスの品質に関する合意を明文化します。

・可用性、レスポンスタイム、サポート対応時間などの項目を含めます。

#### SLOの設定

・SLAを達成するための内部的な目標として、より厳しい基準のSLOを設定します。

・主要な指標（KPI）ごとに具体的な数値目標を設定します。

・例：可用性99.99%、95パーセンタイルレスポンスタイム200ms以内、など

#### モニタリングとSLO

・設定したSLOを継続的にモニタリングし、達成状況を把握します。

・SLOを満たせない状況が続く場合は、原因分析と改善策の検討を行います。

#### 定期的な見直し

・技術の進歩や利用者のニーズの変化に応じて、SLAとSLOを定期的に見直します。

・見直しの結果は、API利用者に適切に通知し、必要に応じて合意を更新します。

# API開発の進め方

APIの利用を促進するためには、提供しているAPIが広く開発者などに認知され、開発を支援するドキュメントやテスト環境などが用意されていることが重要になります。

ここでは、API開発のオープン規格であるOpen API Specification（OAS）を活用した、API開発作業について説明します。

## API提供の告知

各府省Webサイトや政策目的別Webサイトなどのトップページにおいて、API提供の告知、紹介ページや開発者向け情報・ツールへのリンクを掲載し、情報を提供します。

加えて、APIの効果的な提供と利用促進のために、包括的な開発者ポータルの構築と運用を推奨します。開発者ポータルは、API利用者にとって重要な情報源となり、APIの採用と正しい使用を促進する中心的な役割を果たします。

### ポータルの基本機能

API一覧：提供されている全APIのカタログを掲載し、各APIの概要、用途、アクセス方法を明示します。

ドキュメント：OpenAPI Specを利用した各APIの仕様書、エンドポイントの説明、パラメータ一覧、レスポンス形式などを提供します。またSwagger UIなどを活用し、APIを直接試せる環境を提供します。

サンプルコード：主要なプログラミング言語やフレームワークでのAPI利用例を提供します。

SDKとライブラリ：可能な場合、様々な言語やプラットフォーム向けのSDKやクライアントライブラリを提供します。

### ユーザー管理とアクセス制御

ユーザー管理：API利用者の登録システムを設け、ユーザーを識別可能なIDやキーなどの発行・管理を行います。

アクセス制御：APIごとのアクセス権限を設定し、適切なアクセス制御を実施します。

### FAQとサポートチャンネル

FAQ：頻繁に寄せられる質問とその回答を整理して掲載します。

フォーラムやコミュニティ：開発者同士が情報交換や質問を行える場を提供します。

サポートチャンネル：技術的な問い合わせに対応するためのチャンネル（メール、チャットなど）を設けます。

### ダッシボードと使用量制限の表示

ダッシボード：API利用者が自身のAPI使用状況を確認できるダッシボードを提供します。

使用量制限の表示：各APIの使用量制限とその現在の状況を表示します。

### 更新情報と通知システム

更新情報：APIの更新、新機能の追加、非推奨化などの情報を適時に公開します。

通知システム：重要な更新や障害情報を登録ユーザーに通知する仕組みを設けます。

### セキュリティガイドラインと利用規約

セキュリティガイドライン：APIを安全に使用するためのベストプラクティスを提供します。

利用規約：APIの利用条件、データの取り扱い、免責事項などを明確に示します。

### フィードバック機能と改善プロセス

フィードバック機能：API利用者からの意見や要望を収集する仕組みを設けます。

改善プロセス：収集したフィードバックを元に、継続的にAPIとポータル自体を改善します。

## オープン規格を活用したAPIの開発（WebAPI）

Open API Initiative（※）は、APIを標準化することで、APIを適切に作成し、APIを簡易に利用できることを目指している団体です。当団体では、API開発フレームワーク「Swagger」をベースに作成されたオープン規格「OAS」を提供しています。

※The Linux Foundationの協力の元、主要ICT企業で構成される団体

OASは誰でも自由に使うことができ、APIを積極的に利用している企業・組織では準拠が進んでいます。OASに基づくAPIを構築する際には、APIを受け取る/提供するプログラムの生成、ドキュメント自動生成、テストフォームの自動生成など、API開発時の工程を効率化できるツールを無料で利用できます。

行政機関においてもOASを採用し、その標準に沿うことで、開発者のAPI仕様の共通的な理解、作成にかかる負荷を減らし、開発作業が効率化されることが期待できます。

OASは検討、改訂が継続的に行われています。更新情報の定期的な確認を推奨します。 OASでは下表のようなツールが提供されています。

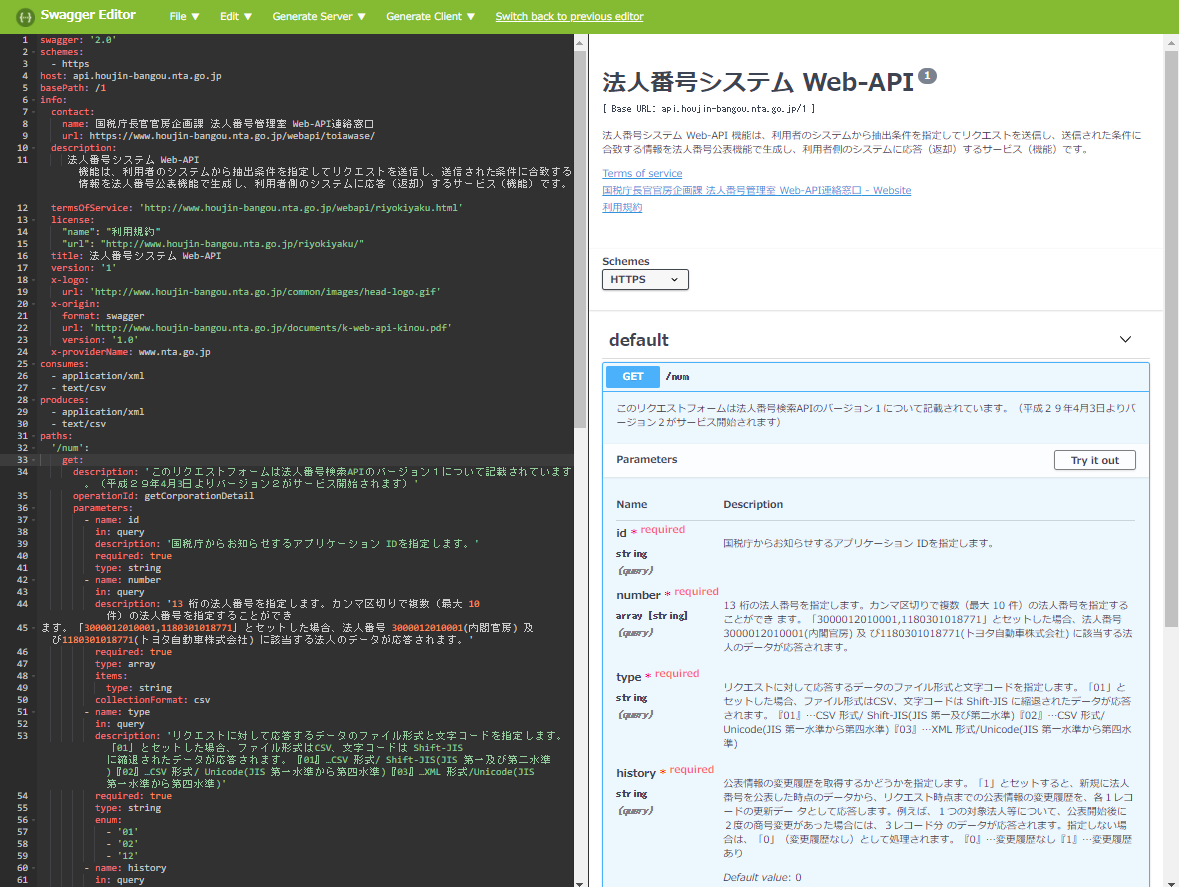
表 ４－１　OASを利用することが出来る主要なツール

|  |  |
| --- | --- |
| ツール名 | 概要 |
| Apicurio Studio | ・ブラウザベースのグラフィカルなAPI設計ツール  ・直感的なUIを通じてOAS準拠のAPI定義を作成・編集できる  ・バージョン管理機能を内蔵し、API設計の更新履歴を追跡可能  ・他のAPIツールとの連携が容易で、設計したAPIを直接他のツールにエクスポート可能 |
| Postman | ・API開発とテストのための総合的なプラットフォーム  ・OAS形式のAPIドキュメントのインポートとエクスポートをサポート  ・APIリクエストの送信、レスポンスの検証、テストスクリプトの作成が可能  ・チーム間でのAPI定義やテストケースの共有機能を提供 |
| Swagger Editor | ・OASに準拠したSwaggerファイルを作成するためのエディタ  ・フレームワークに従って入力すれば、OASに沿った整理ができ、要件漏れ等を防げる。  ・レスポンスデータの出力テストがその場ででき、APIにより提供するデータを確認することができる。 |
| Swagger UI | ・Swaggerファイルを元にして、APIのドキュメントとテストフォームを作成することができる。  ・API構築完了時にテストフォームからのリクエストを元に、動作を確認することができる。 |
| OpenAPI Generator | ・OAS 3.0に完全対応し、より多くのプログラミング言語とフレームワークをサポート  ・クライアントSDK、サーバスタブ、APIドキュメントの生成が可能  ・カスタムテンプレートによる柔軟なコード生成をサポート |
|  |  |

これらのツールを活用することでOASの標準に沿ったAPI仕様書を作ることができます。

また、Swaggerファイルを公開することで、独自にカスタマイズしたAPIを作成し、利用することができます。

図 ４－１　 Swagger Editorの利用イメージ

****

※：Swagger Editor ONLINE EDITORhttp://editor.swagger.io/#/

図 ４－２　 Swagger Editorの利用イメージ



※：Swagger UI LIVE DEMOhttp://petstore.swagger.io/

以下のように、OASに準拠しているAPIであれば、OAS準拠のAPIを公開しているポータルにてSwaggerファイルを公開することができます。API利用者は、同一開発フレーム上で、登録されたAPIを同じ使い方で利用できます。

図 ４－３　OAS準拠のAPIを公開しているポータル例



※：[左]Any APIhttps://any-api.com/ 　[右]APIs.GURUhttps://apis.guru/

## ドキュメントの公開

API提供に当たり、公開を推奨するドキュメントを下表に示します。

Swagger UIを利用することで、自動でAPI仕様のドキュメントを生成できます。

表 ４－２　公開を推奨するドキュメント

|  |  |
| --- | --- |
| ドキュメント名称 | 含めるべき項目 |
|  |  |
| API概要  （「4参考情報」に記入例を示す。） | ・データ内容の概要  当該APIで取得することができるデータの内容を記載する。（API利用者が概要を把握するドキュメントであり、利用者の目に留まりやすい公開方法（Webサイトのトップ等）を推奨。）  ・提供方式  RESTやSOAPなど方式を記載する。  ・エンドポイント  エンドポイントのURIを記載する。  ・OAS定義ファイルの公開場所  OAS定義ファイルを公開する場所のURLを示す。OASに準拠する場合のみ記述する。  ・リクエストフォーマット形式  リクエストする方法（メソッドなど）を示す。  ・レスポンスフォーマット形式  レスポンスのデータ形式を示す。  ・TLS(SSL)サポート  TLS(SSL)サポートの対応状況を記載する。  ・利用申請  利用時の申請の有無を記載する。  ・認証機能  どの認証機能を用いているかを記載する。  ・利用制限  利用制限をかけている場合には、その内容を具体的に示す。  ・開発者専用サイト  開発者のAPI利用者を支援するサイト（テストフォーム提供、サンプル公開など）の有無を記載する。有る場合にはURLを示す。  ・利用規約  提供データの利用規約を示す。詳細を説明するページなどがあれば、そのURLを示す、など |
| API仕様書 | ・API機能  取り扱うデータや操作内容など機能を説明  ・API利用方法  エンドポイント、エンコード、認証機能、利用制限など  ・エラーコード  ・リクエスト  パラメータ、データ形式・データ項目の説明、サンプルなど  ・レスポンス  データ形式・データ項目の説明、サンプルなど  ・提供データに関する説明  データ更新日、提供元、データ内容、更新タイミング（特にデータ提供API）　など |
| 利用規約 | ・コンテンツの利用条件（ライセンスなど）  ・クレジット表示（クレジット表示の要否、内容など）  ・禁止事項  ・免責　など |
| 利用申請 | API利用申請の手順 |
| 利用事例 | APIを用いた事例紹介 |

## テストフォーム及びテスト環境

テストフォームを提供することで、APIの動作仕様を検証しつつ、開発を行うことができます。

なお、Swagger UIを利用すると自動でテストフォームを生成できます。

テストフォームは、入力に手間がかからない工夫がされていれば確認がスムーズに行えます（例えば、デフォルト設定がある項目（言語など）は、自動で入力、指定パラメータ以外の入力不可項目はプルダウンにするなど）。

また、手続APIにおいては、実サービスへ影響を与えないように、テストすることが求められます。そのため、検証環境、テスト用データによる本番環境でのテスト実施等の検討が必要となります。

APIの開発と統合をより効率的に進めるために、APIモックサービスの活用を推奨します。APIモックサービスは、実際のAPIが完成する前に、予定されているAPIの動作をシミュレートすることができます。これにより、フロントエンド開発者はバックエンドAPIの完成を待たずに開発を進めることができ、並行開発が可能になります。APIモックサービスを使用することで、API仕様の早期検証、クライアントアプリケーションの迅速な開発、そして潜在的な問題の早期発見が可能になります。特に、OpenAPI Specification（OAS）に基づいてAPIモックを自動生成できるツールの使用を推奨します。これにより、API仕様の更新に迅速に対応し、常に最新の仕様に基づいたモックを提供することができます。

## サンプルプログラムの提供

ドキュメント等と合わせて、サンプルプログラムをAPI利用者に提供することにより、利用者の利便性を高められます。API利用の促進と開発者の利便性向上を図るため、多言語対応のSDK（Software Development Kit）の提供を推奨します。多言語SDKを提供することで、様々なプログラミング言語やフレームワークを使用する開発者が、より容易にAPIを利用できるようになります。

SDKの開発に当たっては、少なくとも以下の主要な言語やフレームワークをカバーすることを検討してください：Java、Python、JavaScript（Node.js）、C#、Ruby、Go、PHP。各SDKには、API呼び出しの基本機能、認証処理、エラーハンドリング、レート制限の管理などの共通機能を実装し、それぞれの言語やフレームワークの慣習に従ったインターフェースを提供することが重要です。

また、SDKのソースコードをオープンソースとして公開し、GitHub等のプラットフォームで管理することを推奨します。これにより、開発者コミュニティからのフィードバックや貢献を受け入れ、SDKの品質向上と機能拡充を継続的に行うことができます。

## 開発者コミュニティ

インターネット上の掲示板など、API利用者間で情報共有する場を提供することで、利用者同士で疑問を解決する仕組みが出来上がり、API提供者の問合せ対応負荷も削減でき、改善点の発見やナレッジの蓄積効果も期待できます。

この取り組みをさらに強化するために、オンラインフォーラムやチャットツールの活用を推奨します。例えば、GitHub Discussionsを使用してオープンな議論の場を設けたり、SlackやDiscord等のチャットプラットフォームで開発者コミュニティチャンネルを作成したりすることが効果的です。これらのツールを活用することで、リアルタイムのコミュニケーションが可能になり、開発者同士の協力や情報交換がより活発になります。

また、定期的なオンラインミートアップやウェビナーを開催し、新機能の紹介、ベストプラクティスの共有、Q&Aセッションなどを行うことで、コミュニティの活性化と知識の普及を図ることができます。これらの活動を通じて、開発者の声を直接聞き、APIの改善や新機能の開発に反映させることが可能になります。

## FAQ

問合せでよく受ける質問と回答例を整理し、提供します。FAQ対応の効率化と24時間対応を実現するために、AIチャットボットの活用を推奨します。AIチャットボットを導入することで、開発者は時間や場所を問わず、即時に回答を得ることができます。

AIチャットボットの実装に当たっては、既存のFAQデータベースを学習データとして活用し、APIの仕様書やドキュメントも含めた包括的な知識ベースを構築します。AI技術を活用して、開発者の質問を正確に理解し、適切な回答を提供できるよう設計してください。

チャットボットが回答できない複雑な質問や新しい問題については、人間のサポートチームにエスカレーションする仕組みを設け、API利用者の課題を適切に解決できるようにしてください。

# 参考情報

## URIの例

以下にURIの推奨される例、改善すべき例を示します。

図 ５－１　推奨される例

|  |
| --- |
| 本の一覧取得APIの場合:  GET https://api.example.go.jp/v1/magazines.json  本の一覧のうち、範囲を指定して集約・取得する場合：  GET https://api.example.go.jp/v1/magazines?limit=25&offset=50  本の一覧を特定のキーによって指定し取得する場合:  GET https://api.example.go.jp/v1/magazines.json?year=2011&sort=title  GET https://api.example.go.jp/v1/magazines.json?topic=economy&year=2011  特定の本のデータをJSONフォーマットにて返却する場合:  GET https://api.example.go.jp/v1/magazines/1234.json  特定の本に関連する記事を全てJSONフォーマットで返却する場合:  GET https://api.example.go.jp/v1/magazines/1234/articles.json  特定の本に関連する記事を全てXMLフォーマットで返却する場合:  GET https://api.example.go.jp/v1/magazines/1234/articles.xml  特定の本に含まれるデータの中の特定のフィールドのみ、JSONフォーマットで返却する場合：  GET https://api.example.go.jp/ v1/magazines/1234.json?fields=title,subtitle  特定の本に関連する記事を投稿する場合：  POST https://api.example.go.jp/v1/magazines/1234/articles 特定の本に関連する情報を更新する場合： PUT https://api.example.go.jp/v1/magazines/1234/articles/2 特定の本に関連する情報を更新する場合（一部）：  PATCH https://api.example.go.jp/v1/magazines/1234/articles/2 特定の本に関連する記事を削除する場合： DELETE https://api.example.go.jp/v1/magazines/1234/articles |

図 ５－２　改善すべき例

|  |
| --- |
| 名詞が複数系ではない:  GET https://www.example.go.jp/magazine  GET https://www.example.go.jp/magazine/1234  GET https://www.example.go.jp/publisher/magazine/1234  URIに動詞が含まれる:  GET https://www.example.go.jp/magazine/1234/create  データの集約キーがURIに入っている：  GET https://www.example.go.jp/magazines/2011/desc  パラメータがキャメルケースで記述されている：  GET https://api.example.go.jp/v1/magazines/1234.json?createAt=20161204 |

※：White House Web API Standardsを参考に作成

## レスポンスサンプルデータ（WebAPI）

レスポンスのJSONデータのサンプルを示します。

図 ５－３　　GET http://example.go.jp/api/v1/magazines.json

|  |
| --- |
| Response body:  {  "metadata": {  "resultset": {  "count": 2,  "offset": 0,  "limit": 10  }  },  "results": [  {  "id": "1234",  "type": "magazine",  "title": "Public Water Systems",  "tags": [  {"id": "125", "name": "Environment"},  {"id": "834", "name": "Water Quality"}  ],  "created": "1231621302"  },  {  "id": 2351,  "type": "magazine",  "title": "Public Schools",  "tags": [  {"id": "125", "name": "Pre-school"},  ],  "created": "126251302"  }  ]  } |

※：White House Web API Standards https://github.com/WhiteHouse/api-standards

## ドキュメント「API概要情報」の例

以下に、RESAS APIにおける、API概要情報の記入例を示します。

なお、RESAS APIはOASに準拠していないため、OAS定義ファイルの公開場所に関する記載は含んでいません。

表 ５－１　API概要情報の記入例（内閣府RESAS API）

|  |  |
| --- | --- |
| 項目 | 記入例 |
| データ内容概要 | 地域別の産業・農林水産業・観光・人口など統計データ |
| 提供方式 | REST |
| エンドポイント | [https://opendata.resas-portal.go.jp/api/{Version}/](https://opendata.resas-portal.go.jp/api/%7bVersion%7d/) |
| リクエスト  フォーマット形式 | URI パラメータ（GET） |
| レスポンス  フォーマット形式 | JSON |
| TLS(SSL)サポート | 対応 |
| 利用制限 | 呼び出し回数制限あり  上限：2000回/日 |
| 利用申請 | あり |
| 認証機能 | APIキー |
| 利用制限 | コール回数上限：2000回/日 |
| 開発者専用サイト | 無し |
| 利用規約 | 出典明記により、編集・加工も含め自由に利用可 |

※：内閣府RESAS APIhttps://opendata.resas-portal.go.jp/

参考文献一覧

参考とした他APIガイド、及び本ガイドブック内で参照した規格や取組などを下表に示します。

表 ５－２　参考とした他APIガイド

|  |  |
| --- | --- |
| 文献名 | URL |
| White House Web API Standards | <https://github.com/WhiteHouse/api-standards> |
| 18F API Standards | <https://github.com/18F/api-standards> |
| Application programming interfaces (APIs)(GOV.UK) | <https://www.gov.uk/service-manual/technology/application-programming-interfaces-apis> |
| API Design Guide(GOV.AU) | <http://apiguide.readthedocs.io/en/latest/index.html> |
| Web API DesignCrafting Interfaces that Developers Love | <https://pages.apigee.com/rs/apigee/images/api-design-ebook-2012-03.pdf> |

表 ５－３　本ガイドブック内で参照した企画や取り組みなど

|  |  |
| --- | --- |
| 文献名 | URL |
| RFC6570  URI Template | <https://tools.ietf.org/html/rfc6570> |
| RFC4627  The application/json Media Type for JavaScript Object Notation (JSON) | <https://tools.ietf.org/html/rfc4627> |
| W3C Extensible Markup Language (XML) 1.0 (Fifth Edition) | <http://www.w3.org/TR/xml/> |
| RFC4180  Common Format and MIME Type for Comma-Separated Values (CSV) Files | <https://tools.ietf.org/html/rfc4180> |
| RFC7946  The GeoJSON Format | <https://tools.ietf.org/html/rfc7946> |
| Open Geospatial Consortium | <http://www.opengeospatial.org/standards/kml> |
| RFC5545「Internet Calendaring and Scheduling Core Object Specification」 | <https://tools.ietf.org/html/rfc5545> |
| RFC6350「vCard Format Specification」 | <https://tools.ietf.org/html/rfc6350> |
| JIS縮退マップについて  （ページ下部の項目） | <https://www.houjin-bangou.nta.go.jp/download/> |
| JSON Hypertext Application Languagedraft-kelly-json-hal-08 | <https://tools.ietf.org/html/draft-kelly-json-hal-08> |
| ISO 639「Language codes」 | <http://www.iso.org/iso/home/standards/language_codes.htm> |
| ISO 4217「Currency codes」 | <http://www.iso.org/iso/home/standards/currency_codes.htm> |
| ISO 3166「Country codes」 | <http://www.iso.org/iso/home/standards/country_codes.htm> |
| JISX0401「都道府県コード」 | <http://www.jisc.go.jp/>  でJISX0401を検索。 |
| JISX0402「市町村コード」 | <http://www.jisc.go.jp/>  でJISX0402を検索。 |
| ISO22324「Societal security」 | <http://www.iso.org/iso/catalogue_detail.htm?csnumber=50061> |
| Security Research Group  「Let's Encrypt」 | https://letsencrypt.org/ja/ |
| OpenID Connect | <http://openid.net/> |
| OpenID Japan | <http://www.openid.or.jp/document/index.html#op-doc-openid-connect> |
| W3C Date and Time Formats  (ISO8601より） | https://www.w3.org/TR/NOTE-datetime |
| 共通語彙基盤 | https://imi.go.jp/goi/ |
| 文字情報基盤 | https://imi.go.jp/mj/ |
| RFC7231  HTTPステータスコード | <https://tools.ietf.org/html/rfc7231> |
| RFC7807  エラーメッセージ | <https://tools.ietf.org/html/rfc7807> |
| Cross-Origin ResourceSharing  W3C Recommendation | <https://www.w3.org/TR/cors/> |
| Open API Initiative | <https://www.openapis.org/> |
| IPA  ウェブサイトにおける脆弱性解説 | <http://www.ipa.go.jp/security/vuln/vuln_contents/xss_solution.html> |
| IPA  安全なウェブサイトの作り方 | <http://www.ipa.go.jp/security/vuln/websecurity.html> |
| IPA  セキュア・プログラミング講座 | <http://www.ipa.go.jp/security/awareness/vendor/programmingv2/index.html> |
| OWASP  REST Security Cheat Sheet | <https://www.owasp.org/index.php/REST_Security_Cheat_Sheet> |

表 ５－４　技術用語説明

|  |  |
| --- | --- |
| 用語 | 概要 |
| HTTPプロトコル | インターネット上で、Webクライアントからリクエストを送り、それを受けてWebサーバからレスポンスを返すことで、データの送受信を行う規約。サーバに要求する動作を定めたメソッドが定められており、GET/POST/PUT/DELETEなどがある。 |
| REST  (本ガイドブック内) | Representational State Transferの略称。HTTPプロトコルに基づいて、URIを指定しリソースを取得する設計様式を指す。本ガイドブックでは、Pragmatic RESTとする。 |
| SOAP | Simple Object Access Protocolの略称。HTTPプロトコル等に基づいて、XMLデータのやり取りを通じ連携を行う方式。「WS-\*」と総称される様々なプロトコルが用意されている。 |
| URI | Uniform Resource Identifierの略称。インターネット上で、リソースの場所などを表現する識別子。 |
| エンドポイント | ネットワークに接続された端末のアドレス。本ガイドブックでは、APIへアクセスするためのアドレスを示す。 |
| ベースURI | URIのうち、指定するリソースによって変わらない共通部分。 |
| ページネーション | データを小さく分割した単位及びそのアクセス先をページと呼ぶ。全データではなく特定のデータを要求する場合、欲しいデータ（ページ）を指定することとなる。 |
| HTTPステータスコード | HTTPプロトコルに従ったWebサーバからのレスポンスにおいて、その通信結果を表す3桁のコード |
| TLS(SSL) | Secure Sockets Layerの略称。インターネット上で通信データを暗号化するための技術。HTTPにおける通信にTLS(SSL)を適用した場合、その通信をHTTPS通信と呼ぶ。 |
| APIキー | APIの利用に必要な暗号鍵のこと。API利用者ごとに発行し、不正なアクセスを防ぐとともに、API呼び出し状況の把握にも使われる。 |
| OpenID Connect | Webサーバ上のリソースへのアクセス権限を管理し、利用者がクライアントにID/PWを教えることなく、アクセスを許可するための仕組み。OAuth2.0を拡張した認証プロトコル。 |
| クロスサイトスクリプティング | Webページの入力フォームなどから不正なスクリプトを挿入し、そのWebページを閲覧したブラウザ上で、スクリプトを実行させる攻撃。スクリプトにより、利用者の分からないところで、別のサーバに通信を行うこともある。 |
| クロスドメイン | １つのWebサイト内でアクセス先のドメインが複数に跨っている状態。 |
| CORS対応 | Cross-Origin Resource Sharingの略称。Webブラウザがオリジンサーバ（主となるオリジナルコンテンツを保有しているサーバ）以外のサーバからデータを取得できるようにする対応。多くのWebブラウザでは、クロスサイトスクリプティングを防ぐために、別のサーバへアクセスしない仕組みとなっている。 |
| DoSアタック | Denial of Serviceアタックの略称。インターネット上から、サーバに過大な負荷を掛けるために、大量アクセスを行う攻撃。サーバが落ちたり、応答速度が遅くなる。 |
| SQLインジェクション | アプリケーションが想定しないSQL文を入力し、データベースの内容を変更したり削除する攻撃。 |
| クロスサイトリクエストフォージェリ | 攻撃者が作成したWebサイトに訪れた利用者が、知らない間に、自動で他のリソースへアクセスし不正な行為をしてしまう攻撃。 |
| クリックジャギング | 攻撃者が作成したWebサイト上で、ある箇所をクリックさせ、別のWebサイトに関する設定情報などを変更させる攻撃。 |
| トランザクション処理 | 複数の処理セットを一つの単位として処理すること。具体的には、全ての工程が正常に実行された場合にのみ、データベースへの書き込みなど、他へ影響を与える処理を実行する。途中で異常があった場合には、全てを最初の状態に戻す。他からの書き換えが処理中に行われないよう、ロックする（自ら以外変更不可とする）ことなどが挙げられる。 |
| インタフェース  （本ガイドブック内） | APIの仕様、つまりAPIを介した通信方法、又は、仕様に基づき実装された外部と通信する接点。 |